
Rational Unified Process Kruchten

Guide to the Unified Process featuring UML, Java and Design Patterns
Applying UML and Patterns Training Course
IBM Rational Unified Process Reference and Certification Guide
Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development: 3rd Edition
The IBM Rational Unified Process for System Z
The Rational Unified Process Made Easy
Learning UML 2.0
Documenting Software Architectures
The Unified Modeling Language User Guide
Software Leadership
Just Enough Software Architecture
Scaling Software Agility
Systems Engineering with SysML/UML
Explanatory Model Analysis
Business Agility and Information Technology Diffusion
Introduction au Rational Unified Process
UML 2 and the Unified Process
The Rational Unified Process
The Rational Unified Process Made Easy
Managing Technical Debt
Object-Oriented Analysis and Design for Information Systems
Software Project Management
The Rational Unified Process
The Rational Unified Process an Introduction Second Edition(□□□□)
The Rational Unified Process
Implementing the IBM Rational Unified Process and Solutions
Rational Unified Process. Introduzione
Managing Software Requirements
Agility and Discipline Made Easy
Managing the Testing Process
Visual Modeling with Rational Rose 2002 and UML
Building J2EE Applications with the Rational Unified Process
Der rational unified process
The Process of Software Architecting
Software Engineering Process with the UPEDU
Software Development for Small Teams
Software Architecture
Applied Software Architecture

Adopting the Rational Unified Process
The Unified Software Development Process

Rational Unified Process Kruchten Downloaded from ftp.bonide.com by guest

EATON ARIANA

Guide to the Unified Process featuring UML, Java and Design Patterns CRC Press

Object-Oriented Analysis and Design for Information Systems clearly explains real object-oriented programming in practice. Expert author Raul Sidnei Wazlawick explains concepts such as object responsibility, visibility and the real need for delegation in detail. The object-oriented code generated by using these concepts in a systematic way is concise, organized and reusable. The patterns and solutions presented in this book are based in research and industrial applications. You will come away with clarity regarding processes and use cases and a clear understanding of how to expand a use case. Wazlawick clearly explains clearly how to build meaningful sequence diagrams. Object-Oriented Analysis and Design for Information Systems illustrates how and why building a class model is not just placing classes into a diagram. You will learn the necessary organizational patterns so that your software architecture will be maintainable. Learn how to build better class models, which are more maintainable and understandable. Write use cases in a more efficient and standardized way, using more effective and less complex diagrams. Build true object-oriented code with division of responsibility and delegation.

Applying UML and Patterns Training Course International Technical Support Organization IBM

- Helps organizations tackle the complexity of implementation and begin seeing immediate return on their significant RUP investment - Another in a successful line of books from authors at Rational/IBM, and the latest in the acclaimed Object Technology Series - Significant co-marketing opportunities with Rational/IBM
IBM Rational Unified Process Reference and Certification Guide Addison-Wesley Professional

A classic treatise that defined the field of applied demand analysis, *Consumer Demand in the United States: Prices, Income, and Consumption Behavior* is now fully updated and expanded for

a new generation. Consumption expenditures by households in the United States account for about 70% of America's GDP. The primary focus in this book is on how households adjust these expenditures in response to changes in price and income. Econometric estimates of price and income elasticities are obtained for an exhaustive array of goods and services using data from surveys conducted by the Bureau of Labor Statistics, providing a better understanding of consumer demand. Practical models for forecasting future price and income elasticities are also demonstrated. Fully revised with over a dozen new chapters and appendices, the book revisits the original Taylor-Houthakker models while examining new material as well, such as the use of quantile regression and the stationarity of consumer preference. It also explores the emerging connection between neuroscience and consumer behavior, integrating the economic literature on demand theory with psychology literature. The most comprehensive treatment of the topic to date, this volume will be an essential resource for any researcher, student or professional economist working on consumer behavior or demand theory, as well as investors and policymakers concerned with the impact of economic fluctuations.

Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development: 3rd Edition "O'Reilly Media, Inc."

With its clear introduction to the Unified Modeling Language (UML) 2.0, this tutorial offers a solid understanding of each topic, covering foundational concepts of object-orientation and an introduction to each of the UML diagram types.

The IBM Rational Unified Process for System Z Pearson Education
The authors explain the underlying software development principles behind the RUP, and guide readers in its application in their organization.

The Rational Unified Process Made Easy Addison-Wesley Professional

Le succès spectaculaire d'UML ne doit pas faire oublier qu'il ne s'agit que d'un langage de modélisation graphique, dont la vocation n'est pas de couvrir tous les aspects du génie logiciel. Complément idéal d'UML, un processus de développement logiciel

tel que le RUP (Rational Unified Process) a précisément pour but de spécifier les différentes phases d'un projet, de définir les tâches de chacun des intervenants, et de contrôler les coûts, les délais et la qualité de l'application logicielle produite. Le RUP est un processus de développement logiciel élaboré et commercialisé par la société Rational Software. Il se présente sous forme d'un guide méthodologique au format HTML, couplé à une base de connaissances et capable de s'interfacer avec divers outils d'expression des besoins, de modélisation UML, d'automatisation des tests, de gestion de configuration, de production de documents, de gestion de projet, etc. Ecrit par le principal concepteur du RUP chez Rational Software, cet ouvrage présente dans une première partie les différentes composantes du processus, ainsi que les grands principes qui le sous-tendent : développement itératif et incrémental, gestion dynamique de l'expression des besoins, architecture à base de composants, contrôle continu de la qualité, etc. La deuxième partie de l'ouvrage décrit les principaux enchaînements d'activités qui composent chaque itération du processus : expression des besoins, analyse et conception, génération de code, test et déploiement gestion de configuration. Un dernier chapitre explique comment mettre en œuvre et personnaliser le Rational Unified Process. Ce livre est destiné à toutes les personnes impliquées dans des projets de développement logiciel : chefs de projet et consultants, analystes et concepteurs, ingénieurs système, développeurs, ingénieurs méthode et qualité, responsables des tests, rédacteurs techniques, etc.

Learning UML 2.0 Addison-Wesley Professional
Presents an approach to software architecture that takes organizational issues into consideration. The approach uses a series of five principles--vision, rhythm, anticipation, partnering, and simplification--to reveal hidden risks and opportunities of software architecture. Complementing these principles are criteria, patterns, and antipatterns. The criteria help assess how well each principle is being performed currently, and the patterns and antipatterns provide guidance on how to apply the principles.
c. Book News Inc.

Documenting Software Architectures Addison-Wesley Professional

UML, the Universal Modeling Language, was the first programming language designed to fulfill the requirement for "universality." However, it is a software-specific language, and does not support the needs of engineers designing from the broader systems-based perspective. Therefore, SysML was created. It has been steadily gaining popularity, and many companies, especially in the heavily-regulated Defense, Automotive, Aerospace, Medical Device and Telecomms industries, are already using SysML, or are planning to switch over to it in the near future. However, little information is currently available on the market regarding SysML. Its use is just on the crest of becoming a widespread phenomenon, and so thousands of software engineers are now beginning to look for training and resources. This book will serve as the one-stop, definitive guide that provide an introduction to SysML, and instruction on how to implement it, for all these new users. *SysML is the latest emerging programming language--250,000 estimated software systems engineers are using it in the US alone! *The first available book on SysML in English *Insider information! The author is a member of the SysML working group and has written sections of the specification *Special focus comparing SysML and UML, and explaining how both can work together

The Unified Modeling Language User Guide Prentice Hall
 "Companies have been implementing large agile projects for a number of years, but the 'stigma' of 'agile only works for small projects' continues to be a frequent barrier for newcomers and a rallying cry for agile critics. What has been missing from the agile literature is a solid, practical book on the specifics of developing large projects in an agile way. Dean Leffingwell's book *Scaling Software Agility* fills this gap admirably. It offers a practical guide to large project issues such as architecture, requirements development, multi-level release planning, and team organization. Leffingwell's book is a necessary guide for large projects and large organizations making the transition to agile development." —Jim Highsmith, director, Agile Practice, Cutter Consortium, author of *Agile Project Management* "There's tension between building software fast and delivering software that lasts, between being ultra-responsive to changes in the market and maintaining a degree of stability. In his latest work, *Scaling Software Agility*, Dean Leffingwell shows how to achieve a

pragmatic balance among these forces. Leffingwell's observations of the problem, his advice on the solution, and his description of the resulting best practices come from experience: he's been there, done that, and has seen what's worked." —Grady Booch, IBM Fellow Agile development practices, while still controversial in some circles, offer undeniable benefits: faster time to market, better responsiveness to changing customer requirements, and higher quality. However, agile practices have been defined and recommended primarily to small teams. In *Scaling Software Agility*, Dean Leffingwell describes how agile methods can be applied to enterprise-class development. Part I provides an overview of the most common and effective agile methods. Part II describes seven best practices of agility that natively scale to the enterprise level. Part III describes an additional set of seven organizational capabilities that companies can master to achieve the full benefits of software agility on an enterprise scale. This book is invaluable to software developers, testers and QA personnel, managers and team leads, as well as to executives of software organizations whose objective is to increase the quality and productivity of the software development process but who are faced with all the challenges of developing software on an enterprise scale.

[Software Leadership](#) Addison-Wesley Professional

This concise book offers a quick introduction to the concepts, structure, content, and motivation of the Rational Unified Process. This revolutionary software development process provides a disciplined approach to assigning, managing, and completing tasks within a software development organization and is the first development process to exploit the full capabilities of the industry-standard Unified Modeling Language. The Rational Unified Process is unique in that it captures many of the proven best practices in modern software development and presents them in a form that can be tailored to a wide range of projects and organizations. In this book, you will discover: what the Rational Unified Process is - and what it is not; the concepts used in the Rational Unified Process, as well as its structure; the best practices that have been synthesized into this process; and how this process can provide the guidance you need for your specific project responsibilities.

[Just Enough Software Architecture](#) Addison-Wesley Professional
 Software and project management consultant Murray Cantor

discusses how to be a good manager and how to build a competitive software team. The text is intended to be accessible to managers with little software background as well as those with extensive experience. A sampling of topics includes software architecture, developing products, improving the efficiency of the organization, the Rational Unified Process, and team leadership. c. Book News Inc.

[Scaling Software Agility](#) Elsevier

I highly recommend this book for anyone who's ever tried to implement RUP on a small project. Pollice and company have demystified and effectively scaled the process while ensuring that its essence hasn't been compromised. A must-have for any RUPster's library! Chris Soskin, Process Engineering Consultant, Toyota Motor Sales Do you want to improve the process on your next project? Perhaps you'd like to combine the best practices from the Rational Unified Process (RUP) and from agile methodologies (such as Extreme Programming). If so, buy this book! *Software Development for Small Teams* describes an entire software development project, from the initial customer contact through delivery of the software. Through a case study, it describes how one small, distributed team designed and applied a successful process. But this is not a perfect case study. The story includes what worked and what didn't, and describes how the team might change its process for the next project. The authors encourage you to assess their results and to use the lessons learned on your next project. Key topics covered include: Achieving a balance between people, process, and tools; recognizing that software develo

[Systems Engineering with SysML/UML](#) Addison Wesley Publishing Company

John Hunt's book guides you through the use of the UML and the Unified Process and their application to Java systems. Key topics focus explicitly on applying the notation and the method to Java. The book is clearly structured and written, making it ideal for practitioners. This second edition is considerably revised and extended and includes examples taken from the latest version of Rational Rose and Together. Considers how Agile Modelling fits with the Unified Process, and presents Design Patterns Self contained - covers both the Unified Process and UML in one book Includes real-world case studies Written by an experienced author and industry expert Ideal for students on Software Engineering

courses

Explanatory Model Analysis Pearson Education

bull; Reflects all of the changes that were integrated into RUP v2003-the latest version of the very popular product bull; Learn the key concepts, fundamentals of structure, integral content, and motivation behind the RUP bull; Covers all phases of the software development lifecycle -from concept, to delivery, to revision

Business Agility and Information Technology Diffusion

Springer Science & Business Media

An updated edition of the best tips and tools to plan, build, and execute a structured test operation In this update of his bestselling book, Rex Black walks you through how to develop essential tools and apply them to your test project. He helps you master the basic tools, apply the techniques to manage your resources, and give each area just the right amount of attention so that you can successfully survive managing a test project! Offering a thorough review of the tools and resources you will need to manage both large and small projects for hardware and software, this book prepares you to adapt the concepts across a broad range of settings. Simple and effective, the tools comply with industry standards and bring you up to date with the best test management practices and tools of leading hardware and software vendors. Rex Black draws from his own numerous testing experiences-- including the bad ones, so you can learn from his mistakes-- to provide you with insightful tips in test project management. He explores such topics as: Dates, budgets, and quality-expectations versus reality Fitting the testing process into the overall development or maintenance process How to choose and when to use test engineers and technicians, contractors and consultants, and external test labs and vendors Setting up and using an effective and simple bug-tracking database Following the status of each test case The companion Web site contains fifty tools, templates, and case studies that will help you put these ideas into action--fast!

Introduction au Rational Unified Process Pearson Education India
The Authoritative, Best-Practice Guide to Improving Development Processes with IBM® Rational Unified Process® (RUP®) This book delivers all the knowledge and insight you need to succeed with the IBM Rational Unified Process and Solutions. Joshua Barnes presents a start-to-finish, best-practice roadmap to the complete implementation cycle of IBM RUP—from projecting ROI and making

the business case through piloting, implementation, mentoring, and beyond. Drawing on his extensive experience leading large-scale IBM RUP implementations and working with some of the industry's most recognized thought leaders in the Software Engineering Process world, Barnes brings together comprehensive “lessons learned” from both successful and failed projects. You'll learn from real-world case studies, including actual project artifacts. Whether you're an executive, software professional, or consultant, this book will help you continuously improve the maturity of your development processes—and reap the benefits: better quality, faster delivery, and more business value. After reading this book you will be able to · Get past the myths of software process improvement to focus on what's truly practical · Identify and evaluate your best candidate process solutions · Objectively project the ROI achievable with IBM RUP and IBM Rational solutions · Develop funding models, business cases, and executive support · Recruit, staff, organize, and motivate your implementation team · Plan for effective integration, process alignment, and change management · Choose the right pilots, learn the right lessons, and develop effective adoption models · Move quickly to successful program-level implementation · Set maturity level goals for process and tool utilization · Map “End States” for both quantity and quality · Plan for training and mentoring—and understand the distinct role of each · Keep the momentum going after your implementation is complete Link to www.upmentors.com, where you can download actual sample implementation documents—not just templates!

www.ibmpressbooks.com Preface xvii Acknowledgments xxiii About the Author xxv Chapter 1: Evaluating Process Solutions 1 Chapter 2: Your First Steps Toward Implementing RUP and IBM Rational Solutions 17 Chapter 3: Assessing Your Organization and Building Your Business Case for Organizational Change 29 Chapter 4: Implementation Team 49 Chapter 5: Setting Up Pre-Integrated and Process-Aligned Tooling 67 Chapter 6: Implementation Approach 75 Chapter 7: Transitioning to a Program Approach 99 Chapter 8: Funding Model 117 Chapter 9: Training and Mentoring Models 131 Chapter 10: Is Your Implementation Complete? 149 Appendix 1: Executive ROI Overview 155 Appendix 2: Detailed Appendix for Executive ROI Overview 159 Appendix 3: Maturity Level Goals—Sample Tasks 167 Index 175

UML 2 and the Unified Process Addison-Wesley Professional
Software architecture—the conceptual glue that holds every phase of a project together for its many stakeholders—is widely recognized as a critical element in modern software development. Practitioners have increasingly discovered that close attention to a software system's architecture pays valuable dividends. Without an architecture that is appropriate for the problem being solved, a project will stumble along or, most likely, fail. Even with a superb architecture, if that architecture is not well understood or well communicated the project is unlikely to succeed. Documenting Software Architectures, Second Edition, provides the most complete and current guidance, independent of language or notation, on how to capture an architecture in a commonly understandable form. Drawing on their extensive experience, the authors first help you decide what information to document, and then, with guidelines and examples (in various notations, including UML), show you how to express an architecture so that others can successfully build, use, and maintain a system from it. The book features rules for sound documentation, the goals and strategies of documentation, architectural views and styles, documentation for software interfaces and software behavior, and templates for capturing and organizing information to generate a coherent package. New and improved in this second edition: Coverage of architectural styles such as service-oriented architectures, multi-tier architectures, and data models Guidance for documentation in an Agile development environment Deeper treatment of documentation of rationale, reflecting best industrial practices Improved templates, reflecting years of use and feedback, and more documentation layout options A new, comprehensive example (available online), featuring documentation of a Web-based service-oriented system Reference guides for three important architecture documentation languages: UML, AADL, and SySML
The Rational Unified Process Pearson Education
Explanatory Model Analysis Explore, Explain and Examine Predictive Models is a set of methods and tools designed to build better predictive models and to monitor their behaviour in a changing environment. Today, the true bottleneck in predictive modelling is neither the lack of data, nor the lack of computational power, nor inadequate algorithms, nor the lack of flexible models. It is the lack of tools for model exploration

(extraction of relationships learned by the model), model explanation (understanding the key factors influencing model decisions) and model examination (identification of model weaknesses and evaluation of model's performance). This book presents a collection of model agnostic methods that may be used for any black-box model together with real-world applications to classification and regression problems.

The Rational Unified Process Made Easy Addison-Wesley Professional

A Comprehensive Process for Defining Software Architectures That Work A good software architecture is the foundation of any successful software system. Effective architecting requires a clear understanding of organizational roles, artifacts, activities performed, and the optimal sequence for performing those activities. With *The Process of Software Architecting*, Peter Eeles and Peter Cripps provide guidance on these challenges by covering all aspects of architecting a software system, introducing best-practice techniques that apply in every environment, whether based on Java EE, Microsoft .NET, or other technologies. Eeles and Cripps first illuminate concepts related to software architecture, including architecture documentation and reusable assets. Next, they present an accessible, task-focused guided tour through a typical project, focusing on the architect's role, with common issues illuminated and addressed throughout. Finally, they conclude with a set of best practices that can be applied to

today's most complex systems. You will come away from this book understanding The role of the architect in a typical software development project How to document a software architecture to satisfy the needs of different stakeholders The applicability of reusable assets in the process of architecting The role of the architect with respect to requirements definition The derivation of an architecture based on a set of requirements The relevance of architecting in creating complex systems *The Process of Software Architecting* will be an indispensable resource for every working and aspiring software architect—and for every project manager and other software professional who needs to understand how architecture influences their work.

Managing Technical Debt Addison-Wesley Professional

"This is an incredibly wise and useful book. The authors have considerable real-world experience in delivering quality systems that matter, and their expertise shines through in these pages. Here you will learn what technical debt is, what is it not, how to manage it, and how to pay it down in responsible ways. This is a book I wish I had when I was just beginning my career. The authors present a myriad of case studies, born from years of experience, and offer a multitude of actionable insights for how to apply it to your project." -Grady Booch, IBM Fellow Master Best Practices for Managing Technical Debt to Promote Software Quality and Productivity As software systems mature, earlier

design or code decisions made in the context of budget or schedule constraints increasingly impede evolution and innovation. This phenomenon is called technical debt, and practical solutions exist. In *Managing Technical Debt*, three leading experts introduce integrated, empirically developed principles and practices that any software professional can use to gain control of technical debt in any software system. Using real-life examples, the authors explain the forms of technical debt that afflict software-intensive systems, their root causes, and their impacts. They introduce proven approaches for identifying and assessing specific sources of technical debt, limiting new debt, and "paying off" debt over time. They describe how to establish managing technical debt as a core software engineering practice in your organization. Discover how technical debt damages manageability, quality, productivity, and morale—and what you can do about it Clarify root causes of debt, including the linked roles of business goals, source code, architecture, testing, and infrastructure Identify technical debt items, and analyze their costs so you can prioritize action Choose the right solution for each technical debt item: eliminate, reduce, or mitigate Integrate software engineering practices that minimize new debt *Managing Technical Debt* will be a valuable resource for every software professional who wants to accelerate innovation in existing systems, or build new systems that will be easier to maintain and evolve.